RESEARCH ARTICLE OPEN ACCESS

REACTIVE CODE GENERATION FOR MODULAR WEB ENGINEERING (MWE) FRAMEWORK

Mensah Y. A*.,Agbaje M., Izang A., Ajayi O. F,Bamidele O., Amusa A. I.,

[1]Babcock University, School of Computing and Engineering Science
mensahy@babcock.edu.ng
[2]Babcock University, School of Computing and Engineering Science
agbajem@babcock.edu.ng
[3]Babcock University, School of Computing and Engineering Science
izanga@babcock.edu.ng
[4]Babcock University, School of Computing and Engineering Science
ajayioluwa@babcock.edu.ng
[5]Babcock University, School of Computing and Engineering Science
bamideleoluw@babcock.edu.ng
[6]Babcock University, School of Computing and Engineering Science
amusaa@babcock.edu.ng

Abstract

Automatic generation of textual artefacts for web engineering frameworks (including code, documentation, configuration files, build scripts, etc.) from models in a software development process through the application of model-to-text (M2T) transformation is a common MDE activity. Despite the importance of M2T transformation, contemporary M2T languages lack support for developing transformations that scale with the size of the input model for web Engineering. As MDWE is applied to systems of increasing size and complexity, a lack of scalability in M2T transformation languages hinders industrial adoption. The main objective of this research is to propose a reactive code generation model for modular web engineering to generate textual artefacts. eclipse IDE and Epsilon object language are used in transforming software models to text thereby generating YAML files. The YAML file is converted to a blueprint and finally translated by the Laravel blueprint engine to a modular software application. The proposed model will help software developers quickly and accurately build software systems.

Keywords: Model-To-Text (M2T), Web Engineering Frameworks, Model-Driven Web Engineering (MDE), Yet Another Markup Language, Epsilon Object Language (EOL), Laravel Blueprint.

1.1 Introduction

Software development has become a complex and challenging issue due to constant changes in requirements[1]. This has made it difficult for developers to quickly and accurately build software systems. The use of models has greatly been embraced over the last decade due to the fact that models are used to provide abstractions for systems under study; it has also provided a common vision and knowledge among technical and non-technical stakeholders thereby promoting direct communication among them [2]. This has made models be seen as not just artefacts for the documentation but also as major artefacts in software engineering processes.

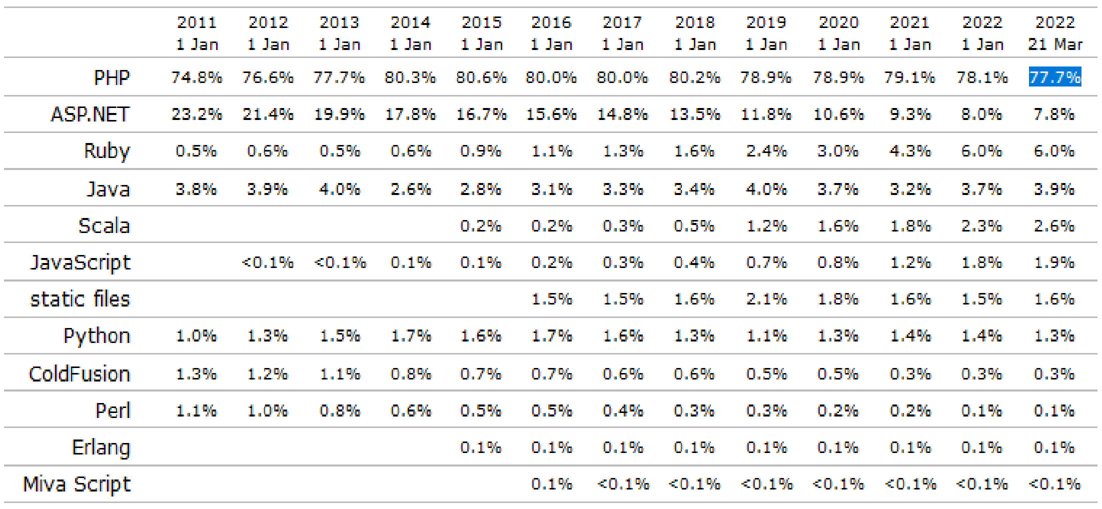
This, in a broader sense, can be referred to as Model Driven Engineering (MDE). MDE according to [3][4] is a paradigm that increases the productivity and automation of software development by adopting models as the primary and major artefacts in the development of software systems. It aims to develop software by transforming models into executable software code [5].

The Model Driven Engineering (MDE) approach has been proposed in the literature to increase productivity and reduce software development costs. The MDE deals with complexity by using models as the main artefacts of the software development process. Despite the numerous benefits of MDE, a major challenge has been the scalability of MDE processes which includes reusability, modularity, and most importantly efficient propagation of changes between artefacts [6][7][8] The introduction Model Driven Engineering (MDE) software Frameworks tend to improve some of the challenges faced. Frameworks haveeliminated the need to write a lot of repetitive code that we will find being used in many different applications[9].The use of a framework is often essential for medium and large-scale developments[10]. Framework which are integrated collection of components that collaborate to produce a reusable architecture for a family of related applications. Design patterns represent solutions to problems that arise when developing software within a particular context, capture the static and dynamic structure, and collaboration in software designshas become the norm of the day among software developers, be it web or desktop technology development. this research tends to look into Model-Driven Web Engineering Frameworks (MDWE)[11]. Web development technologies refer to the multitude of programming languages and tools that are used to produce dynamic and fully-featured websites and applications with front-end and back-end technologies[12]. Front-end technologies are used for the "clientside" of a web application. They're used to develop the interactive components of a website and back-end technologies are used for the "serverside" of a web application.

Web Engineering Frameworks (WEF) are tools and libraries that developers use to make developing in a particular language easier and more efficient. They provide interfaces to access commonly-used functionalities as well as abstractions that make complicated things easier to understand and handle [12]. This technology can be divided into two which are Model Driven and None-Model Driven.

MDWE framework is a framework that supports the Model-View-Controller (MVC). MVC is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller[13]. Each of these components is built to handle specific development aspects of an application. Some of the common frameworks are Laravel, Django, Express.js, ASP.NET Core, Angular, and Zend. This research focuses on the Laravel framework, Yet Another Markup Language, and PHP programming language.

Laravel uses Yet Another Markup Language (YAML) files to create modular processes. YAML is a data serialization language that is often used for writing configuration files. YAML has features that come from Perl, C, XML, HTML, and other programming languages. YAML is also a superset of JSON, so JSON files are valid in YAML. [14]. Also, PHP is the most used web programming language with 77.7% as shown in Figure 1 [15].



	2011 1 Jan	2012 1 Jan	2013 1 Jan	2014 1 Jan	2015 1 Jan	2016 1 Jan	2017 1 Jan	2018 1 Jan	2019 1 Jan	2020 1 Jan	2021 1 Jan	2022 1 Jan	2022 21 Mar
PHP	74.8%	76.6%	77.7%	80.3%	80.6%	80.0%	80.0%	80.2%	78.9%	78.9%	79.1%	78.1%	77.7%
ASP.NET	23.2%	21.4%	19.9%	17.8%	16.7%	15.6%	14.8%	13.5%	11.8%	10.6%	9.3%	8.0%	7.8%
Ruby	0.5%	0.6%	0.5%	0.6%	0.9%	1.1%	1.3%	1.6%	2.4%	3.0%	4.3%	6.0%	6.0%
Java	3.8%	3.9%	4.0%	2.6%	2.8%	3.1%	3.3%	3.4%	4.0%	3.7%	3.2%	3.7%	3.9%
Scala					0.2%	0.2%	0.3%	0.5%	1.2%	1.6%	1.8%	2.3%	2.6%
JavaScript		<0.1%	<0.1%	0.1%	0.1%	0.2%	0.3%	0.4%	0.7%	0.8%	1.2%	1.8%	1.9%
static files						1.5%	1.5%	1.6%	2.1%	1.8%	1.6%	1.5%	1.6%
Python	1.0%	1.3%	1.5%	1.7%	1.6%	1.7%	1.6%	1.3%	1.1%	1.3%	1.4%	1.4%	1.3%
ColdFusion	1.3%	1.2%	1.1%	0.8%	0.7%	0.7%	0.6%	0.6%	0.5%	0.5%	0.3%	0.3%	0.3%
Perl	1.1%	1.0%	0.8%	0.6%	0.5%	0.5%	0.4%	0.3%	0.3%	0.2%	0.2%	0.1%	0.1%
Erlang					0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%	0.1%
Miva Script						0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%

Figure 1: Usage statistics of server-side programming languages for websites[15]**.**

Despite the numerous benefits of MDWE Frameworks, a major challenge has been the scalability of MDWE processes which includes the composition of framework control, composition with legacy components, framework gap, an overlap of framework entities, the composition of entity functionality and most importantly efficient propagation of changes between artefacts [6][7][8]. As models increase due to the complexity of the system they represent, Model-to-Text (M2T) transformation becomes a challenge as the amount of time used in the transformation increases without a clear distinction of model elements that should be excluded from the regeneration process based on either the incremental or batch consistency checking.

Automatic translation of Unified Modelling Language (UML) which is a collection of loosely connected diagram-centric design notations or SIMULINK to working software framework code with the help of code generators is desirable due to the reasonable elimination of errors in the translation process. This can be achieved through batch (on-demand) transformation(s) which are called for by the engineers and/or through live transformations which are triggered automatically by querying specifically the source elements which are relevant for the propagation of the software code in monolithic models – this technique(s) does not take into consideration, modular models.

Research works in literature have shown different ways to combat and reduce the time spent in M2T transformation but these solutions are quite expensive and in turn, have led to some bottlenecks for large monolithic models. To this end, this study intends to implement a transformation technique that will be optimized for large models spread over several interlinked files.

2.0 Literature Review

MDWE frameworks provide techniques to systematically develop queries as well as transformations used in automated code generators to process models. This is evident from the handful of research articles as regards MDWE framework across various academic search engines which will be duly reviewed.

All the literature reviewed developed an incremental M2M transformation framework that allows the propagation of change from the source model to the target model. Altamimi&Petriu, (2017) specifically achieved this by adopting an incremental change propagation (ICP) approach to propagate changes from the UML+MARTE software model to the corresponding LQN mode, where only affected parts of the models were updated when changes were applied to the source model. Another approach adopted by Bergmann et al., (2012) was to introduce a change-driven model transformation by developing a change history model which serves as a log containing the history of model changes, the log recorded timelines between changes and was derived incrementally based on live transformation during model editing which enabled the propagation of changes to the target model at a desired time. The major difference with [18] was that the developed transformation framework can be configured to select translation rules as needed to update the target model while the framework developed by [19] treats transformation as black box- requiring no details of the transformation rules thereby preventing the re-implementation of such transformations.

[20] presented an approach to the instant consistency checking of UML models which quickly, correctly, and automatically decides what consistency rule(s) to evaluate whenever there is a change in a model. This approach assisted engineers in managing the rate of change in large-scale models with thousands of elements due to the fact that engineers can decide when to re-evaluate consistency rules and when to display inconsistencies. [21] presented and developed an approach for achieving a fully automated source-incremental M2T transformation through the use of property access traces which provides an efficient means for determining which subset of the transformation is to be re-executed, this in turn generates the required changes in the software codes – this avoids the regeneration of non-affected portions of the source model and thus improves scalability.

Though incremental code generation and consistency checking techniques have proven to be faster than their batch counterparts [20] and have to a great extent minimized the need for redundant computation thereby reducing the execution time of M2T transformation, its benefits disappear for transformations that produce large monolithic files [21]. Besides, the code generation engine needs to be able to tell what changed in the model at a given and precise level which might be quite expensive in its own right in terms of computer resources used and in the long run becomes a bottleneck for large monolithic models. These techniques were not optimized for modular models.

Based on the reviews, it is evident that different M2T transformation models can be adopted for the development of Model-Driven Web Engineering Frameworks (MDWE) solutions with each having its pros and cons. An attempt is therefore made to propose a transformation technique that will be optimized for large models that can spread over a number of interlinked files.

3.0 Model Driven Engineering Overview

Model-driven engineering (MDE) is an iterative and incremental software development process. Supporting the analysis and the verification of software systems developed following the MDE paradigm requires toadopt incrementality when carrying out these crucial tasks in a more optimized way[22].

The first tools to support MDE were the computer-Aided Software Engineering (CASE) tools developed in the 1980s and with several variations of the modeling definitions such as Booch, Rumbaugh, Jacobson, Gane and Sarson, Harel, Shlaer and Mellor, and others were eventually joined creating the Unified Modeling Language (UML)[23]. Unified Modeling Language (UML) is a general-purpose, developmental modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system[24] thereby, promoting the development of models at different levels of abstraction. The higher-level models are (automatically) transformed into lower-level models and finally into the code [25].

Model-driven engineering (MDE) over the last decade has emerged as a successful and widely used approach for developing software systems where models act as the key development artifact. Modeling languages ranging from the standardized

Unified Modeling Language (UML) to domain-specific languages (DSLs) and profiles, such as the SysML, are being used under the MDE umbrella [26]. Also, Models are used for analysis (analogs to the Computational Independent Model (CIM) of Model Driven Architecture (MDA)), design (analogous to the Platform Independent Model (PIM) of MDA), and implementation (analogous to Platform Specific Model (PSM) of MDA) of software artifacts in the software life cycle. Figure 2 shows the architecture of various models of MDA and the transformation among them.

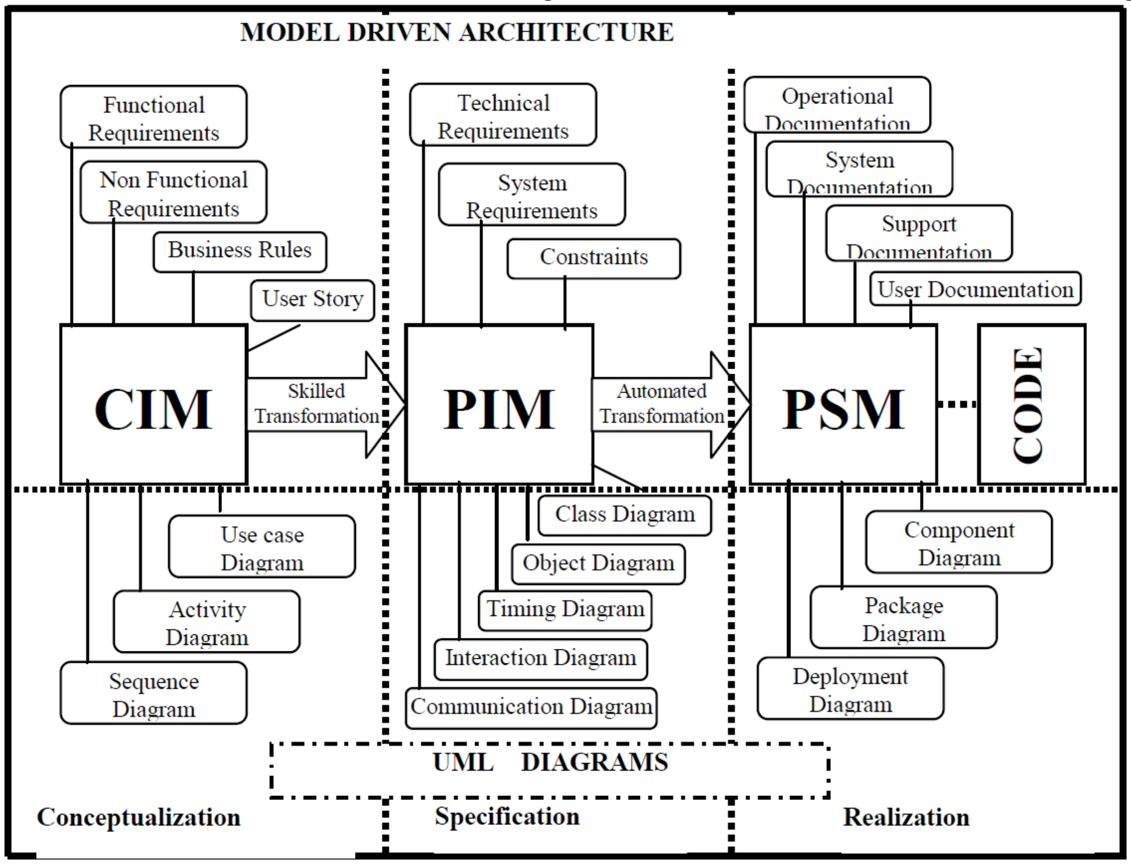


Figure 2: Concepts of Model Driven Architecture.

3.1 Web Engineering Frameworks (WEF)

Web Engineering Frameworks (WAF) is a software framework that is designed to support the development of web applications including web services, web resources, and web APIs. It provides a standard way to build and deploy web applications on the World Wide Web. Web Engineering Frameworks aim to automate the overhead associated with common activities performed in web development. For example, many web frameworks provide libraries for database access, templating frameworks, and session management, and they often promote code reuse [27]. Although they often target the development of dynamic websites, they are also applicable to static websites[28].Types of web engineering framework architectures include Model–view–controller (MVC), Push-based and Pull-based, and Three-tier organization.

3.1.1 Model–view–controller (MVC)

Many frameworks follow the MVC architectural pattern to separate the data model into business rules (the "controller") and the user interface (the "view"). This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied.

3.1.2 Push-based and Pull-based

Most MVC frameworks follow a push-based architecture also called “action-based”. These frameworks use actions that do the required processing, and then “push” the data to the view layer to render the results[29] Django, Ruby on Rails, Symfony, Spring MVC, Stripes, Sails.js, CodeIgniter[30] are good examples of this architecture. An alternative to this is pull-based architecture, sometimes also called "component-based". These frameworks start with the view layer, which can then "pull" results from multiple controllers as needed. In this architecture, multiple controllers can be involved with a single view. Lift, Tapestry, JBoss Seam, Jakarta Server Faces, and Wicket are examples of pull-based architectures. Play, Struts, RIFE, and ZK have support for both push- and pull-based application controller calls.

3.1.3 Three-tier organization

In a three-tier organization, applications are structured around three physical tiers: client, application, and database[31]. The database is normally an RDBMS. The application contains the business logic, runs on a server, and communicates with the client using HTTP[32]. The client on web applications is a web browser that runs HTML generated by the application layer[33]. The term should not be confused with MVC, where, unlike in three-tier architecture, it is considered a good practice to keep business logic away from the controller, the "middle layer"[31].

3.2 Types ofWeb Engineering Frameworks (MDWE)

Examples of web engineering framework includes: Angular, ASP.NET Core, Express.js, Django, Laravel, and Zend.

i. **Angular Framework:**Angular is a TypeScript-based, free, and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS[34].

ii. **ASP.NET Core Framework:** ASP.NET Core is a free and open-source web framework and successor to ASP.NET, developed by Microsoft. It is a modular framework that runs on both the full .NET Framework, on Windows, and the cross-platform .NET. However, ASP.NET Core version 3 only works on .NET Core, dropping support for the .NET Framework[35].

iii. **Express.js Framework:**Express.js, or simply Express, is a back-end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js[36].

iv. **Django Framework:**Django is a free and open-source, Python-based web framework that follows the model–template–views architectural pattern[37]. It is maintained by the Django Software Foundation, an independent organization established in the US as a 501 non-profit[38]. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself[39].

v. **Zend Framework**

Laminas Project (formerly Zend Framework or ZF) is an open-source, object-oriented web application framework implemented in PHP 7 and licensed under the New BSD License[40]. The framework is a collection of professional PHP[41]based packages[42]. The framework uses various packages by the use of Composer as part of its package dependency managers MVC implementation in Laminas has five main areas. The router and dispatcher functions to decide which controller to run based on data from the URL, and controller functions in combination with the model and view to develop and create the final web page[42]

vi. **Laravel Framework:** Laravel is a PHP-based development framework that supports the MVC architecture and offers high-speed web development it has an end-to-end development framework that comes with various features like Middleware, ORM, and session management. it's simple to use and helps with a swift web development process[43].

4.0 Modularityin Web Engineering

Modularity refers to the extent to which a software or Web application may be divided into smaller modules. Software modularity indicates that the number of application modules is capable of serving a specified business domain. Modularization offers the advantages to cover these requirements of flexibility and fast integration as well as scalability and adaptability.

4.1 Properties of Modularityin Web Engineering

There are five major properties of modularity which are.

1. **Modular Decomposability**: Decomposability simply means breaking down something into smaller pieces. Modular decomposability means breaking down the problem into different sub-problems in a systematic manner. Solving a large problem is difficult sometimes, so the decomposition helps in reducing the complexity of the problem, and the sub-problems created can be solved independently. This helps in achieving the basic principle of modularity.
2. **Modular Composability:** Composability simply means the ability to combine modules that are created. It's the principle of system design that deals with how two or more components are related or connected. Modular composability means assembling the modules into a new system which means connecting the combined components into a new system.
3. **Modular Understandability:**Understandability simply means the capability of being understood, the quality of comprehensible. Modular understandability means making it easier for the user to understand each module so that it is very easy to develop software and change it as per requirement. Sometimes it's not easy to understand the process models because of their complexity and their large size structure. Using modularity understandability, it becomes easier to understand the problem efficiently without any issues.
4. **Modular Continuity:** Continuity simply means an unbroken or consistent or uninterrupted connection for a long period without any change or being stopped. Modular continuity means making changes to the system requirements that will cause changes in the modules individually without causing any effect of change in the overall system or software.
5. **Modular Protection:** Protection simply means to keep something safe from any harm, to protect against any unpleasant means or damage. Modular protection means to keep safe the other modules from abnormal conditions occurring in a particular module at run time. The abnormal condition can be an error or failure also known as a run-time error. The side effects of these errors are constrained within the module.

5.0 Reactive Code

Reactive coding describes a design paradigm that relies on asynchronous programming logic to handle real-time updates to otherwise static content. It provides an efficient means of the use of automated data streams to handle data updates to content whenever a user inquires.

5.1 Features of Reactive Code for Modular Web Engineering

Reactive Coding is a programming paradigm that deals with Concurrent and Multi-Threading processes. The features that are the baseline requirements for Reactive Codes in Modular Web Engineering are identified below.

1. **Asynchronous & Non-blocking:**A thread that is assigned to a request will not wait till the response comes back instead an event will be sent to the blocker (DB server/Web Service) & informs that the thread is not going to wait for the response.
2. **Background Synchronization:** Ability to synchronize data in the background.
3. **Functional style of coding:**Ability to implement call-back handlers' methods which take care of the asynchronous execution.
4. **Data flow as event-driven stream:**Ability to notify the user of whatever happened at the producer end based on how the user configures the producer.
5. **Backpressure on data streams:**Backpressure on the flow of data helps a user to send the response in a serialized manner.

5.2 Components of Reactive Code for Modular Web Engineering

There are four major properties of modularity which are

1. **Eclipse:** is an integrated development environment (IDE) used in computer programming[44] It contains a base workspace and an extensible plug-in system for customizing the environment. It is the second-most-popular IDE for Java development, and, until 2016, was the most popular[45].
2. **EOL**: is the core expression language of Epsilon, and the foundation for task-specific languages for tasks such as model-validation, model-to-text transformation, model-to-model transformation, and model migration.
3. **Blueprint:** is an open-source tool for rapidly generating multiple Laravel components from a single, human-readable definition.
4. **YAML:** is a human-readable data-serialization language. It is commonly used for configuration files and in applications where data is being stored or transmitted.

6.0 Proposed Solution

The proposed solution to the challenges and negative impact on software development complexity brought about by a change in requirement is a reactive code generation model for modular web engineering. The model shown in Figure 3 can be adopted by any software development organization to help in reducing the time spent in developing software.

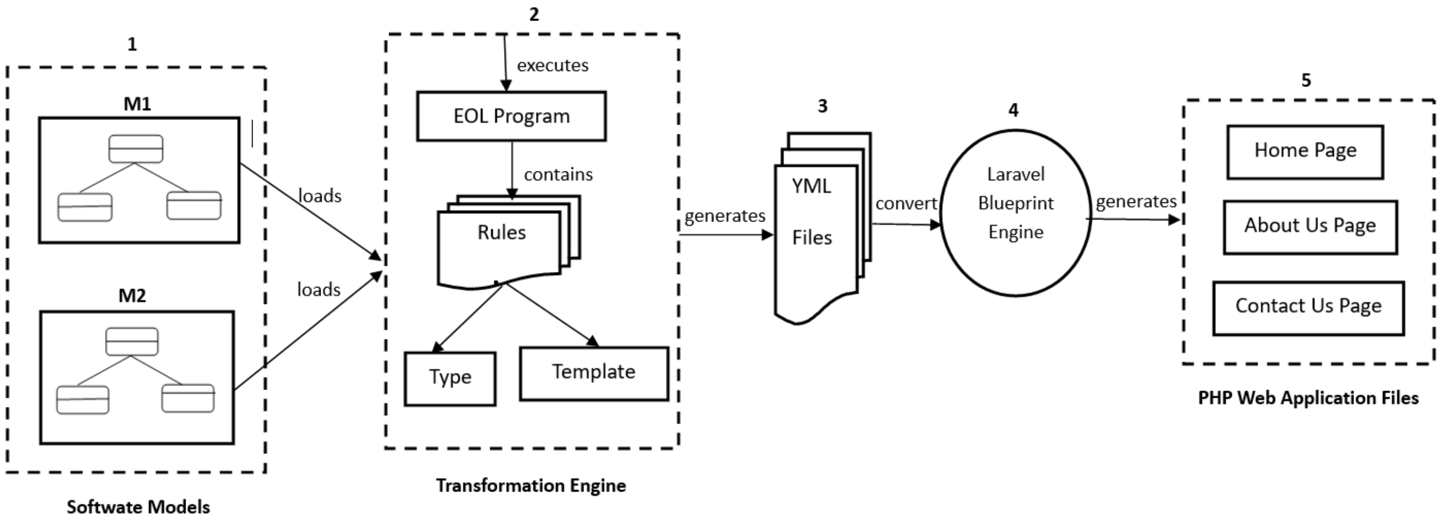


Figure 3: Reactive Code Generation for Modular Web Engineering Model (Researcher's Model)

Based on Figure 1, the model flow is detailed below.

Step 1: The user drawsa UML model diagram using Eclipse IDE

Step 2: After a successful model drawing, the model is transformed from model to text using epsilon object language (EOL). Also, type and template rules are created to generate a YAML file.

Step 3: After generating the YAML file, the YAML file is converted to a blueprint.

Step 4: Once the blueprint is generated the Laravel blueprint engine translates it to the desired language (PHP, C#, Phyton, or JavaScript) and splits the print into separate folders as a modular web application (model, view, and controller)

Step 5: After a successful transformation of the web application file form a full software.

6.0 Conclusion

Software development has become a complex and challenging issue due to a constant change in requirements which has made it difficult for developers to quickly and accurately build software systems. The reactive code generation model for modular web engineering has brought a revolution in web development and this needs to be embraced. Adopting reactive code generation model technology for modular web engineering frameworks will make software engineers develop software systems quickly and accurately. This research recommends the adoption of the proposed model to ease the development of software applications and facilitate quick production in a software organization.

REFERENCES

[1] J. Sánchez, "Challenges of traditional and agile software processes," *dcc.uchile.cl*, Nov. 2018, Accessed: Oct. 29, 2021. [Online]. Available: https://www.dcc.uchile.cl/TR/2018/TR_DCC-20180719-003.pdf

[2] A. D. S.-C. Languages, S. & Structures, and undefined 2015, "Model-driven engineering: A survey supported by the unified conceptual model," *Elsevier*, Accessed: Oct. 25, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1477842415000408

[3] A. Bucchiarone and J. Cabot, "Grand challenges in model-driven engineering: an

analysis of the state of the research," *Springer*, vol. 19, no. 1, pp. 5–13, 2020, Accessed: Oct. 29, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s10270-019-00773-6

[4] B. Selic, "What will it take? A view on adoption of model-based methods in practice," *Softw Syst Model*, vol. 11, no. 4, pp. 513–526, Oct. 2012, doi: 10.1007/S10270-012-0261-0.

[5] K. Balasubramanian, A. Gokhale, G. K.-Computer, and undefined 2006, "Developing applications using model-driven design environments," *ieeexplore.ieee.org*, Accessed: Oct. 25, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1597085/

[6] B. Ogunyomi, L. M. Rose, and D. S. Kolovos, "Incremental execution of model-to-text transformations using property access traces," *Softw Syst Model*, vol. 18, no. 1, pp. 367–383, Feb. 2019, doi: 10.1007/S10270-018-0666-5.

[7] D. Kolovos and R. Paige, "Scalability: The holy grail of model driven engineering," *ssel.vub.ac.be*, 2008, Accessed: Oct. 25, 2021. [Online]. Available: http://ssel.vub.ac.be/ChaMDE08/_media/chamde2008_proceedingsd121.pdf?id=wsorganisation&cache=cache#page=10

[8] J. Warmer and A. Kleppe, "Building a flexible software factory using partial domain specific models," *dsmforum.org*, vol. 10, no. 6, pp. 1–8, 2006, Accessed: Oct. 25, 2021. [Online]. Available: http://dsmforum.org/events/DSM06/Papers/OOPSLA-DSM-06-Proceedings.pdf#page=22

[9] IBM, "What is software development? | IBM," 2019. https://www.ibm.com/topics/software-development (accessed Oct. 25, 2021).

[10] K. Conboy and N. Carroll, "Implementing Large-Scale Agile Frameworks: Challenges and Recommendations", doi: 10.1109/MS.2018.2884865.

[11] N. Mwendi Edwin, "Software Frameworks, Architectural and Design Patterns," *Journal of Software Engineering and Applications*, vol. 7, pp. 670–678, 2014, doi: 10.4236/jsea.2014.78061.

[12] L. Chantelle, "An Introduction to Web Development Technologies | Tiller Digital," Dec. 20, 2019. https://tillerdigital.com/blog/an-introduction-to-web-development-technologies/ (accessed Oct. 25, 2021).

[13] M. Matthew, "MVC Framework Tutorial for Beginners: What is, Architecture & Example," Oct. 07, 2021. https://www.guru99.com/mvc-tutorial.html (accessed Oct. 26, 2021).

[14] w3schools, "Difference between yaml and yml file extension and comparison," *w3schools*, 2021. https://www.w3schools.io/file/yaml-vs-yml/ (accessed Oct. 29, 2021).

[15] W3Techs, "Usage Statistics and Market Share of Server-side Programming Languages for Websites, November 2021," *Q-Success*, Nov. 01, 2021. https://w3techs.com/technologies/overview/programming_language (accessed Nov. 01, 2021).

[16] T. Altamimi and D. Petriu, "Incremental change propagation from UML software models to LQN performance models," 2017.

[17] G. Bergmann, I. Ráth, G. Varró, and D. Varró, "Change-driven model transformations: Change (in) the rule to rule the change," *Softw Syst Model*, vol. 11, no. 3, pp. 431–461, Jul. 2012, doi: 10.1007/S10270-011-0197-9/METRICS.

[18] S. Johann and A. Egyed, "Instant and incremental transformation of models," *Proceedings - 19th International*

Conference on Automated Software Engineering, ASE 2004, pp. 362–365, 2004, doi: 10.1109/ASE.2004.1342765.

[19] A. Razavi, K. Kontogiannis, C. Brealey, and L. Nigul, "Incremental model synchronization in model driven development environments," *Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '09*, pp. 216–230, 2009, doi: 10.1145/1723028.1723053.

[20] A. Egyed, "Instant consistency checking for the UML," *Proceedings - International Conference on Software Engineering*, vol. 2006, pp. 381–390, 2006, doi: 10.1145/1134285.1134339.

[21] B. Ogunyomi, L. M. Rose, and D. S. Kolovos, "Incremental execution of model-to-text transformations using property access traces," *Softw Syst Model*, vol. 18, no. 1, pp. 367–383, Feb. 2019, doi: 10.1007/S10270-018-0666-5/METRICS.

[22] A. Khalil and J. Dingel, "Optimizing the Symbolic Execution of Evolving Rhapsody Statecharts," *Advances in Computers*, vol. 108, pp. 145–281, Jan. 2018, doi: 10.1016/bs.adcom.2017.09.003.

[23] D. C. Schmidt, "Model-Driven Engineering," *IEEE Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006, doi: 10.1109/MC.2006.58.

[24] M. Thielking, L. la Sala, K. Taylor, J. Mackelprang, and P. Flatau, *Unified Modeling Language User Guide, The*, 2nd ed., no. August. Addison-Wesley, 2005. Accessed: Feb. 19, 2023. [Online]. Available: http://www.informit.com/store/unified-modeling-language-user-guide-9780321267979

[25] I. Rožanc and M. Mernik, "The screening phase in systematic reviews: Can we speed up the process?," *Advances in Computers*, vol. 123, pp. 115–191, Jan. 2021, doi: 10.1016/bs.adcom.2021.01.006.

[26] A. A. Jilani, M. Z. Iqbal, M. U. Khan, and M. Usman, "Advances in Applications of Object Constraint Language for Software Engineering," *Advances in Computers*, vol. 112, pp. 135–184, Jan. 2019, doi: 10.1016/bs.adcom.2017.12.003.

[27] E. M. Maximilien, "Web Services on Rails: Using Ruby and Rails for Web Services Development and Mashups," *IEEE Xplore*, p. 43, Dec. 2006, doi: 10.1109/ICWS.2006.139.

[28] StaticGen, "Top Open-Source Static Site Generators," *StaticGen*, 2006, Accessed: Feb. 20, 2023. [Online]. Available: https://www.staticgen.com/

[29] K. Thomson, *Clarification on MVC= Pull and MVC Push*. 2003. Accessed: Apr. 12, 2023. [Online]. Available: https://www.theserverside.com/discussions/thread/22143.html

[30] Donald J. Brown, "What are the fundamental differences between Struts and JSF - Apache Struts 2 Wiki - Apache Software Foundation," *confluence*, Oct. 26, 2007. https://cwiki.apache.org/confluence/display/WW/What+are+the+fundamental+differences+between+Struts+and+JSF (accessed Apr. 12, 2023).

[31] M. KLIMUSHYN, "Web Application Architecture – Client-Side vs. Server-Side," *Atomic Spin*, Apr. 2015, Accessed: Apr. 12, 2023. [Online]. Available: https://spin.atomicobject.com/2015/04/06/web-app-client-side-server-side/

[32] E. M. Maximilien, "Web Services on Rails: Using Ruby and Rails for Web Services Development and Mashups," *IEEE Xplore*, p. 43, Dec. 2006, doi: 10.1109/ICWS.2006.139.

[33] J. Rao, D. Dimitrov, P. Hofmann, and N. Sadeh, "A Mixed Approach to Semantic Web Service Discovery and Composition," *2006 IEEE International Conference on Web Services (ICWS'06)*, pp. 401–410,

2006, Accessed: Apr. 12, 2023. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4032051

[34] AngularJS, *AngularJS and Angular 2+: a Detailed Comparison*. 2018. Accessed: Feb. 20, 2023. [Online]. Available: https://www.sitepoint.com/angularjs-vs-angular/

[35] ASP.NET, "Announcing ASP.NET Core in .NET 6," *.NET Blog*, Nov. 2021, Accessed: Feb. 20, 2023. [Online]. Available: https://devblogs.microsoft.com/dotnet/announcing-asp-net-core-in-net-6/

[36] ExpressJS, "Express 4.x changelog," *expressjs.com*, 2022, Accessed: Feb. 20, 2023. [Online]. Available: http://expressjs.com/en/changelog/4x.html

[37] H. Adrian and K.-M. Jacob, *The Django Book*. 2020. Accessed: Feb. 20, 2023. [Online]. Available: http://www.djangobook.com/en/2.0/chapter05.html#the-mtv-or-mvc-development-pattern

[38] A. Ravindran, "Django design patterns and best practices : easily build maintainable websites with powerful and relevant Django design patterns," 2022.

[39] Django, "Design Philosophies," *Django*, 2018, Accessed: Feb. 20, 2023. [Online]. Available: https://docs.djangoproject.com/en/2.0/misc/design-philosophies/

[40] Zend, "Introduction to Zend Framework," *ZF Programmer's Reference Guide*, 2009, Accessed: Feb. 20, 2023. [Online]. Available: http://framework.zend.com/manual/en/introduction.html

[41] w3schools, "PHP 5 Tutorial," *www.w3schools.com*, 2020, Accessed: Feb. 20, 2023. [Online]. Available: https://www.w3schools.com/php/

[42] a R. W. C. Zend, "Zend Framework - About," *framework.zend.com*, 2013, Accessed: Feb. 20, 2023. [Online]. Available: https://framework.zend.com/about

[43] M. Kamaruzzaman, "Top 10 In-Demand Web Development Frameworks in 2021 | by Md Kamaruzzaman | Towards Data Science," *towardsdatascience*, Dec. 08, 2020. https://towardsdatascience.com/top-10-in-demand-web-development-frameworks-in-2021-8a5b668be0d6 (accessed Oct. 29, 2021).

[44] zeroturnaround, "IDEs vs. Build Tools: How Eclipse, IntelliJ IDEA & NetBeans users work with Maven, Ant, SBT & Gradle," *zeroturnaround.com*, 2018. https://zeroturnaround.com/rebellabs/ides-vs-build-tools-how-eclipse-intellij-idea-netbeans-users-work-with-maven-ant-sbt-gradle/ (accessed Apr. 12, 2023).

[45] Snyk, "IntelliJ IDEA dominates the IDE market with 62% adoption among JVM developers," *Snyk*, Feb. 2020, Accessed: Apr. 12, 2023. [Online]. Available: https://snyk.io/blog/intellij-idea-dominates-the-ide-market-with-62-adoption-among-jvm-developers/